Session 8

Black Box Testing (2)

Equivalence Partitioning

chengbaolei@suda.edu.cn

Black Box Testing Techniques

- Boundary Value Analysis
- Equivalence Partitioning
- Decision Table
- Cause-effect Graph
- Combinatorial Test

等价关系的定义与实例

定义 设 R 为非空集合上的关系. 如果 R 是自反的、对称的和传递的, 则称 R 为 A 上的等价关系. 设 R 是一个等价关系, 若$<x,y>\in R$, 称 x 等价于y, 记做 $x\sim y$.

实例 设 $A=\{1,2,\ldots,8\}$, 如下定义A上的关系 R:

$$R = \{ <x,y> \mid x,y\in A \wedge x\equiv y(\text{mod } 3) \}$$

其中 $x\equiv y(\text{mod } 3)$ 叫做 x 与 y 模3相等, 即 x 除以3的余数与 y 除以3的余数相等.

等价关系的验证

验证模 3 相等关系 R 为 A 上的等价关系, 因为

$\forall x \in A$, 有 $x \equiv x \pmod 3$

$\forall x, y \in A$, 若 $x \equiv y \pmod 3$, 则有 $y \equiv x \pmod 3$

$\forall x, y, z \in A$, 若 $x \equiv y \pmod 3$, $y \equiv z \pmod 3$,

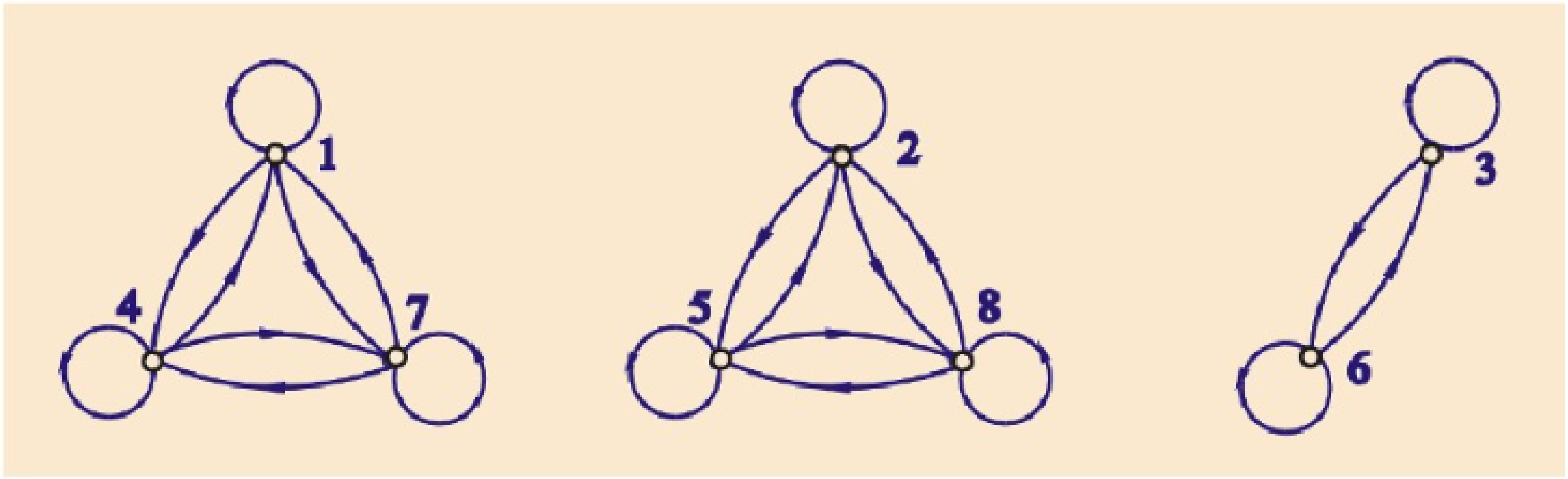
则有 $x \equiv z \pmod 3$

自反性、对称性、传递性得到验证

A上模3等价关系的关系图

设 $A=\{1,2,\ldots,8\}$,

$R=\{ \langle x,y\rangle \mid x,y \in A \wedge x \equiv y \pmod 3) \}$



等价类

定义 设R为非空集合A上的等价关系, $\forall x \in A$， 令

$$[x]_R = \{ y \mid y \in A \wedge xRy \}$$

称 $[x]_R$ 为 x 关于R 的等价类, 简称为 x 的等价类, 简记为$[x]$.

实例 A={ 1, 2, … , 8 }上模 3 等价关系的等价类:

$[1]=[4]=[7]=\{1,4,7\}$

$[2]=[5]=[8]=\{2,5,8\}$

$[3]=[6]=\{3,6\}$

等价类的性质

定理1 设R是非空集合A上的等价关系, 则

(1) $\forall x \in A$, $[x]$ 是A的非空子集.

(2) $\forall x, y \in A$, 如果 $x \, R \, y$, 则 $[x]=[y]$.

(3) $\forall x, y \in A$, 如果 $x \not{R} \, y$, 则 $[x]$与$[y]$不交.

(4) $\cup\{ [x] \mid x \in A\}=A$，即所有等价类的并集就是$A$.

实例

A={ 1, 2, … , 8 }上模 3 等价关系的等价类：

[1]=[4]=[7]={1,4,7}，

[2]=[5]=[8]={2,5,8}，

[3]=[6]={3,6}

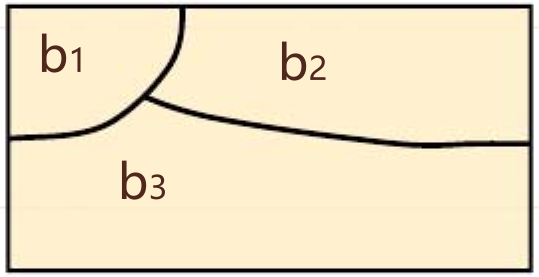
以上3 类两两不交，

$\{1,4,7\}\cup\{2,5,8\}\cup\{3,6\} = \{1,2, \dots ,8\}$

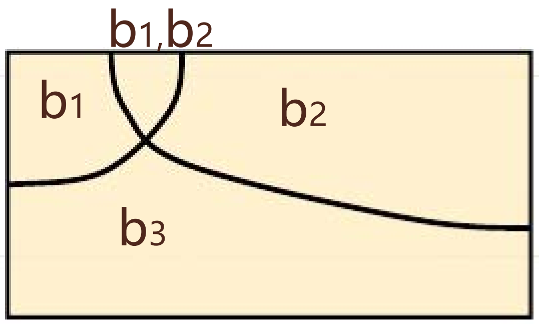
Equivalence Class

The ECs should partition the input domain (the set of all possible input to the program)

1. The ECs should be pairwise disjoint (no overlapping to reduce redundancy in testing)
2. Together the ECs should cover the domain (to ensure the completeness of testing)



A partition of D



Not a partition

Completeness?

Equivalence Partitioning

Equivalence partitioning allows a tester to subdivide the input into a number of equivalence classes (EC).

Each EC defines a set of input such that it is likely to be handled by the system in the same way.

The tester select <u>one</u> test case from each equivalence class.

Motivation

Equivalence Class:

If one test case in an EC detects a defects, all other test cases in the same EC are likely to detect the same defect

If one test case in an EC does not detect a defect,no other test cases in the same EC is likely to detect the defect.

Example: sum of two integers

Consider a program which adds two integers a and b.

Consider the following set of ECs:

$V1=\{(a,b):a>0\}$

$V2=\{(a,b):b>0\}$

$V3=\{(a,b):a<0\}$

$V4=\{(a,b):b<0\}$

Does the ECs partition the input domain?

Improved formulation

VA1={a: a>0}
VA2={a: a=0}
VA3={a: a<0}

VB1={(b: b>0}
VB2={(b: b=0}
VB3={(b: b<0}

Steps of EC partitioningTesting

1.Identify the input/output parameters
  Parameters to a method, data read from file, global variable

2.Identify characteristics for the input/output parameters
  e.g. range of input variables, relationship among variables

3.Each characteristic is partitioned into Equivalence Classes

4.Select a strategy for combinations of equivalence class for different characteristics
  e.g.strong vs weak testing

5.Identify infeasible combinations of equivalence classes

6.Test Case Generation
  For each feasible EC, generate one test case to cover the EC

Example: Interest Rate

A savings account in a bank earns a different rate of interest depending on the balance in the account. If a balance is in the range \$0 up to \$100 has a 3% interest rate,a balance over \$100 and up to \$1000 has a 5% interest rate,and balances of \$1000 and over have a 7% interest rate.

Write a program which inputs the account balance and outputs the amount of interest for a year.

Step1.Identify the input parameters

We only have one variable: account balance

Step2.Identify characteristics for the input parameters

For the account balance parameter, we create partitions based on the range of account balance

Step3.Each characteristic is partitioned into Equivalence Classes

The following partitions are identified:

Valid Partitions:

VP1: 0<=balance<=100
VP2: 100<balance<1000
VP3: balance>=1000

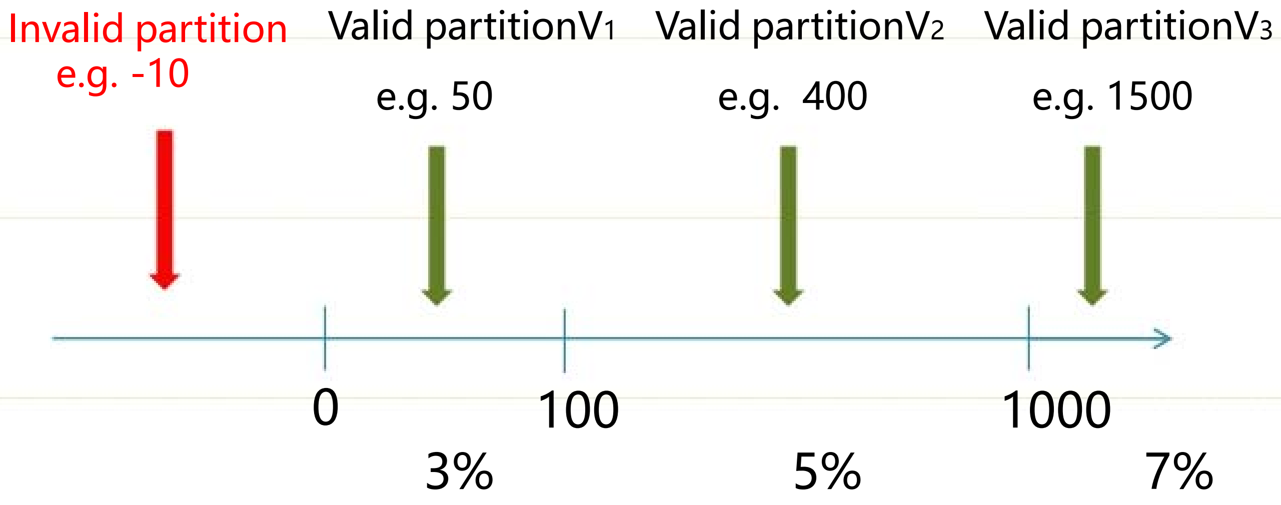
Invalid Partition:

IP1: balance<0

Does the ECs partition the input domain?

Step 4:Test Case Generation

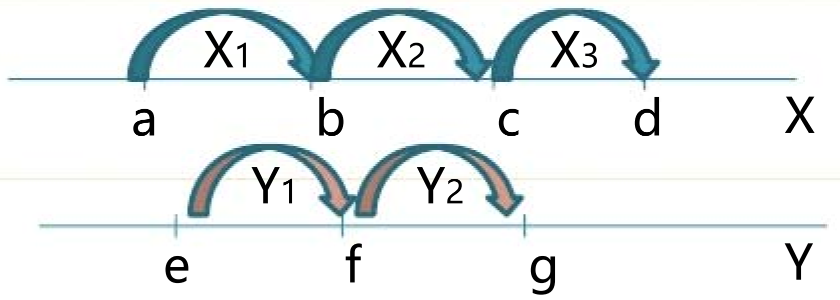
Test Case	EC	Input	Expected output
TC1	{balance: 0<=balance<=100}	Balance=50	$1.5
TC2	{balance: 100<balance<1000}	Balance=400	$20
TC3	{balance: balance>=1000}	Balance=1500	$105
TC4	{balance :balance<0}	Balance=-10	Invalid input



Equivalence Partitioning for multiple variables/characteristics

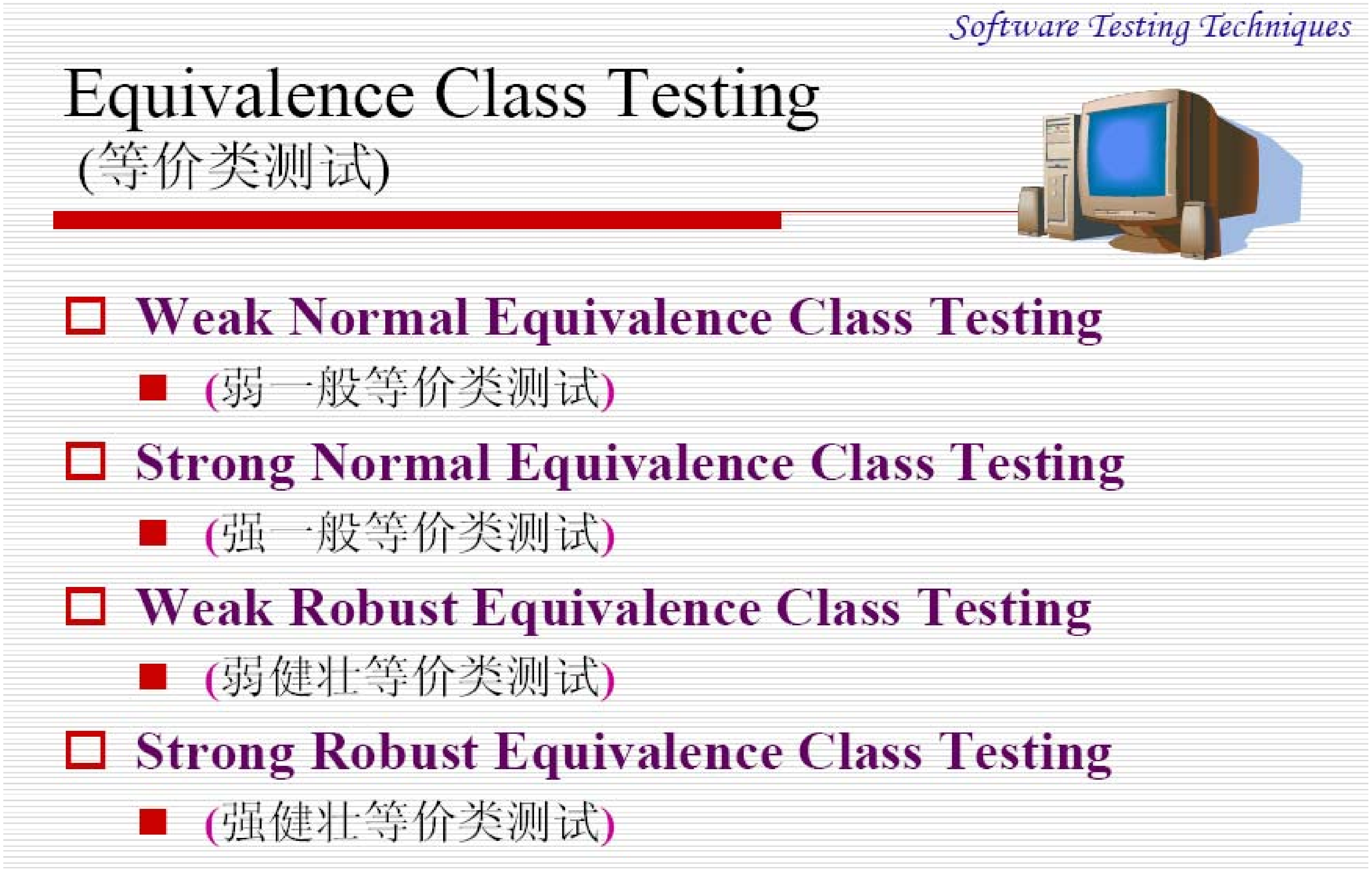
Consider a function f which accepts two integers x and y.

a<=X<=d with intervals [a,b),[b,c),[c,d]
e<=Y<=g with intervals [e,f),[f,g]



VX1={x: a<=x<b}
VX2={x: b<=x<c}
VX3={x: c<=x<=d}

VY1={x: e<=x<f}
VY2={x: f<=x<=g}



Equivalence Class Testing (等价类测试)

- **Weak Normal Equivalence Class Testing**
  - (弱一般等价类测试)
- **Strong Normal Equivalence Class Testing**
  - (强一般等价类测试)
- **Weak Robust Equivalence Class Testing**
  - (弱健壮等价类测试)
- **Strong Robust Equivalence Class Testing**
  - (强健壮等价类测试)

Hybrid approach

Approach for test case selection

Strong ECTesting:

A test case is selected for each element in the set of Cartesian Product of the input variables/characteristics

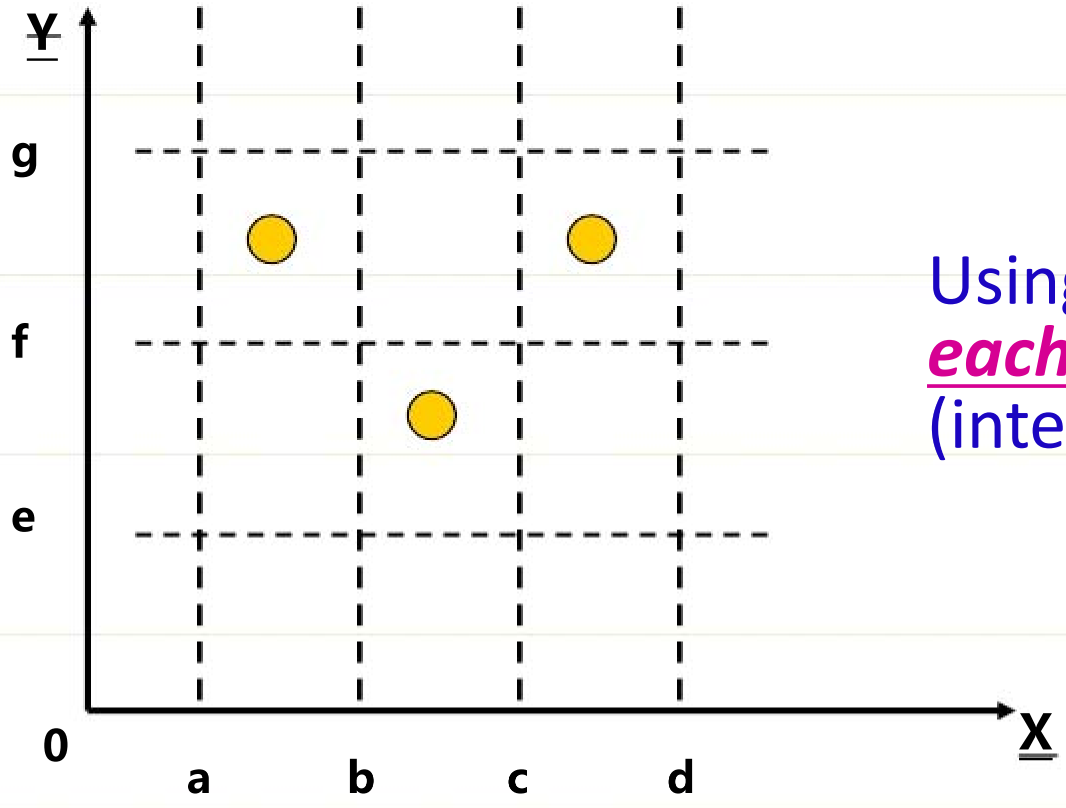
e.g.$EC_x \times EC_y$

$=\{(x_1,y_1),(x_1,y_2),(x_2,y_1),(x_2,y_2),(x_3,y_1),(x_3,y_2)\}$

Weak ECTesting:

A value from each EC appears in at least one test case.Values of different input variables/ characteristics are chosen independent.

Weak Normal Equivalence Testing



Using ***one variable from each equivalence class*** (interval) in a test case.

Useful when the variables/characteristics are independent: If there is fault with handling a certain value $x_i \in X$,we can produce a failure by testing X independent of Y (with any valid value of $y_i \in Y$).

Interaction Fault

An interaction fault is triggered when a certain combinations of t>=1 input value causes the program containing the fault to enter an invalid state,which propagate to a point where it is observable and reveal the fault

Simple Fault (t=1)

A fault can be triggered by some value of an input variable, regardless of the value of other input variables.

Pair-wise interaction fault (t=2)

A fault can be triggered only when two input variables are set to specific value

The definition can be generalized to a t-way interaction fault, where t>=2.

Interaction fault:example 1

```
int  x,  y;
input(x,y);
if  (x>=0)
   output  (2*y);
else
   output  (3*y);

```

Should be x + 1>=0

Assume x,y $\in$ {-2,-1,1,2}

What test case should be used to reveal the fault? Should strong or weak testing be used to detect the fault?

Interaction fault:example 1

```
int  x,  y;
input(x,y);
if  (x>=0)
   output  (2*y);
else
   output  (3*y);
```

Should be x + 1>=0

Assume x,y ∈ {-2,-1,1,2}

The fault is revealed when x = -1 with any value of y. This is an example of simple fault.

To detect the presence of a simple fault in x, we just need to test the different values of x with any value of y. Weak testing is sufficient.

Interaction fault:example 2

```
int  x,  y;
input(x,y);
if  (x  &&  y)
  output(f(x,y));
else
  output  (g(x,y));
```

Should be x || y

Assume x,y ∈ {T,F}

What test case should be used to reveal the fault? Should strong or weak testing be used to detect the fault?

Interaction fault:example 2

```
int  x,  y;
input(x,y);
if  (x  &&  y)
   output(f(x,y));
else
   output  (g(x,y));
```

Should be x || y

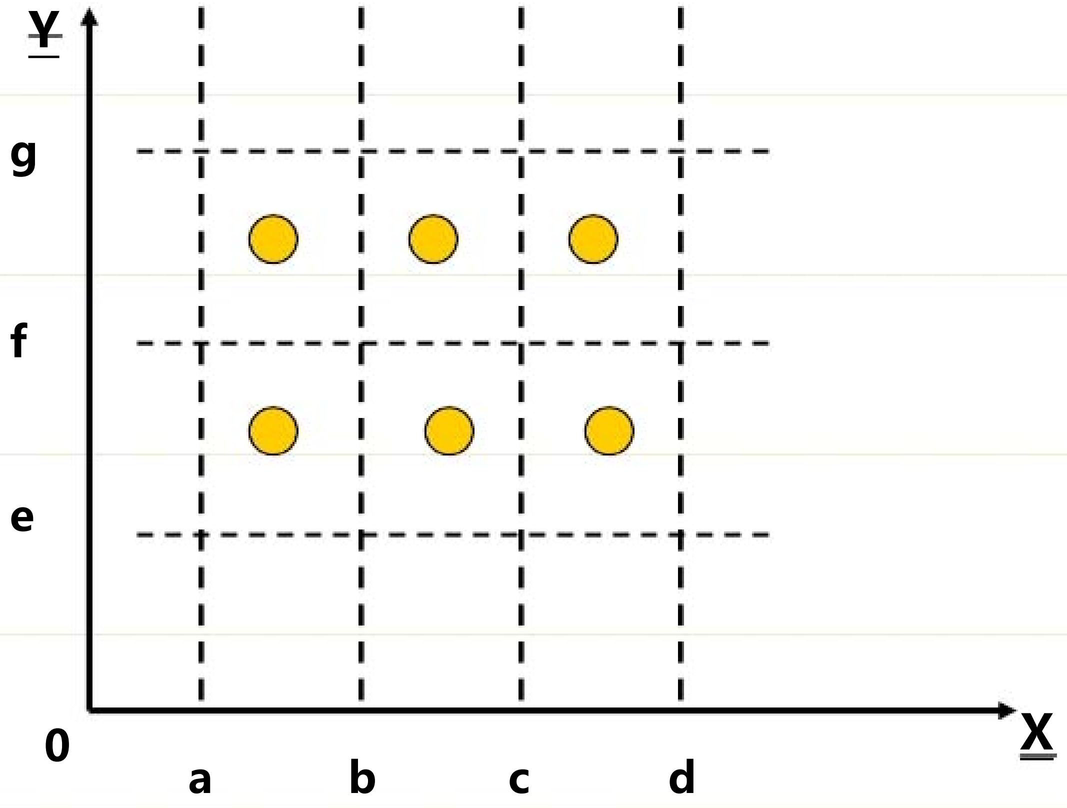
Assume x∈ {T,F} ,y ∈ {T,F}

x	y	x&&y	x||y	Detect the fault?
T	T	T	T	No
T	F	F	T	Yes
F	T	F	T	Yes
F	F	F	F	No

When x is true,the fault can only be detected when y is false (but not when y is true). Weak testing may not be able to reveal the fault (if the inappropriate value for y is chosen).

This is an example of pair-wise interaction fault.

Strong Normal Equivalence Testing



Useful when the variables/characteristics may interact and some faults can only be revealed when certain combinations of values of X and Y appears together.

Limitations of Strong Testing

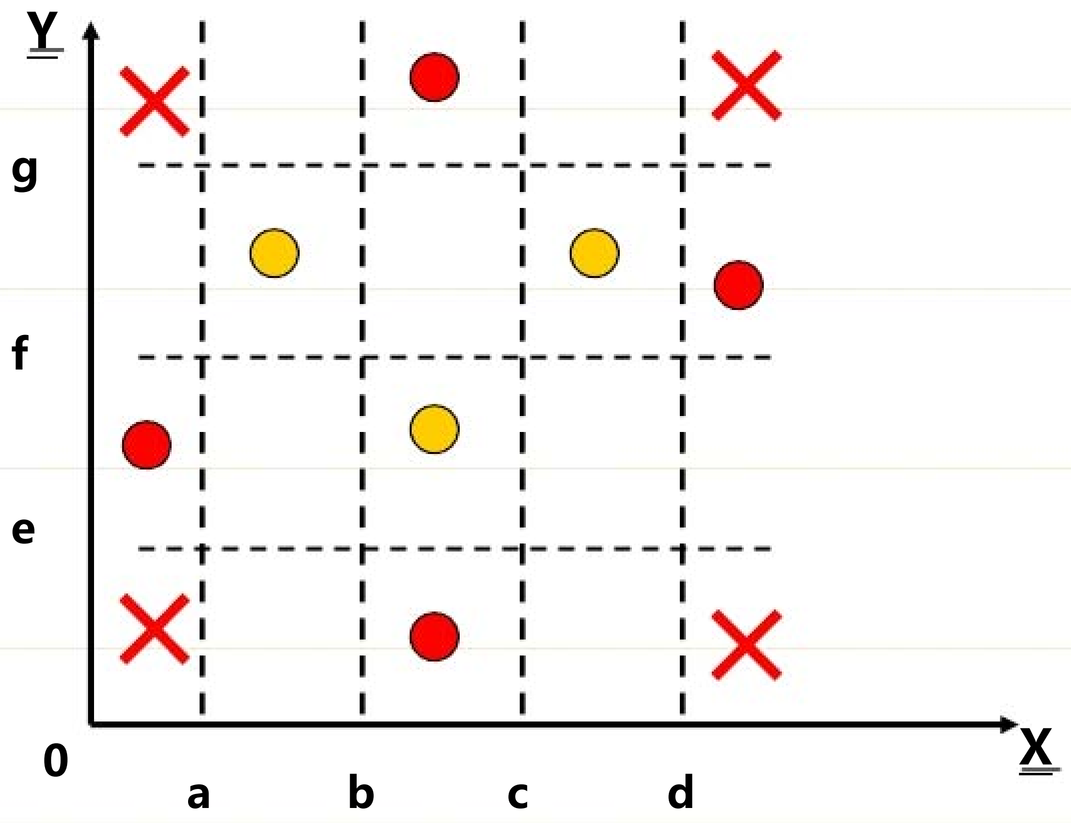
May generate many test cases,some of which may be impossible,redundant,not interesting.

Example: consider the NextDate example

31 June, 30 Feb…

Consider the feasible combinations.

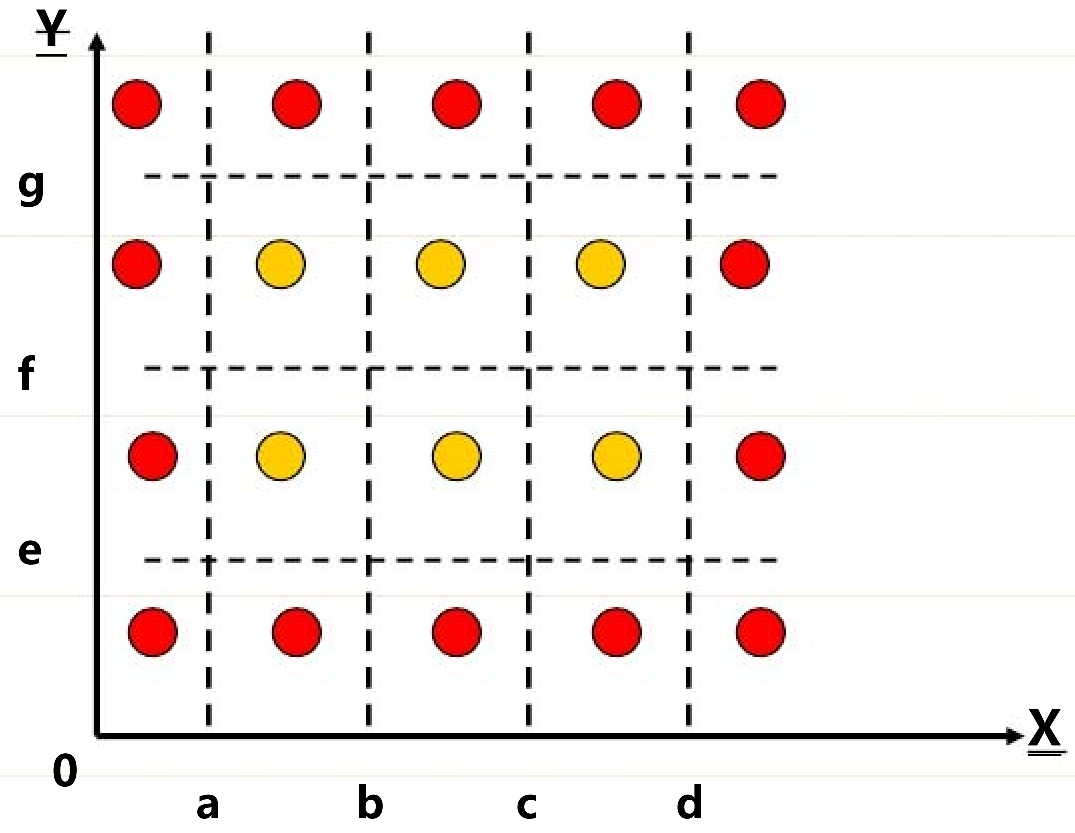
Weak Robust EquivalenceTesting



1. For valid inputs, use one value from each valid class (weak normal equivalence class).

2. For invalid inputs, a test case will have one invalid value and the remaining values will all be valid.

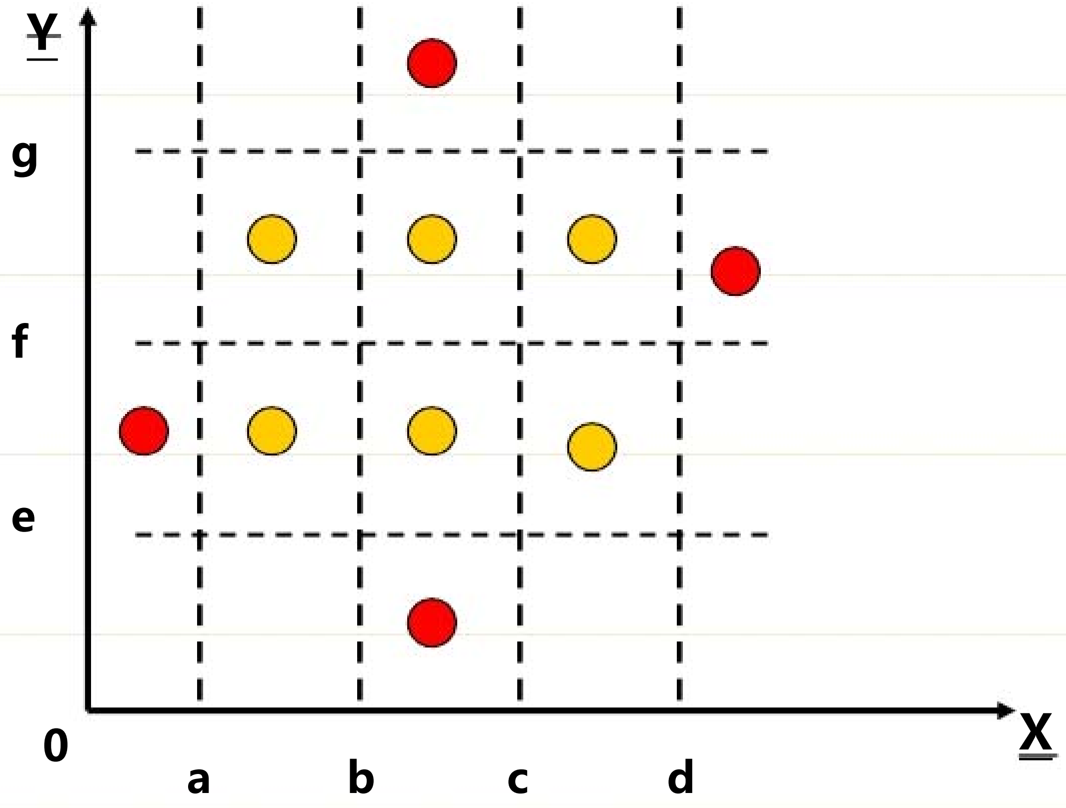
Strong Robust EquivalenceTesting



The robust part comes from consideration of invalid values

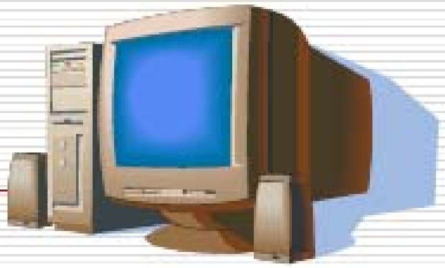
The strong part refers to the multiple fault assumption.

Hybrid approach



Use weak testing for invalid input and strong testing for valid input.
Generate test cases for every combinations of valid ECs (strong testing).
For invalid EC,only combine with one of the valid EC (weak testing).

Equivalence Class Testing
(等价类测试)



- **Weak Normal Equivalence Class Testing**
  - (弱一般等价类测试)
- **Strong Normal Equivalence Class Testing**
  - (强一般等价类测试)
- **Weak Robust Equivalence Class Testing**
  - (弱健壮等价类测试)
- **Strong Robust Equivalence Class Testing**
  - (强健壮等价类测试)

Hybrid approach

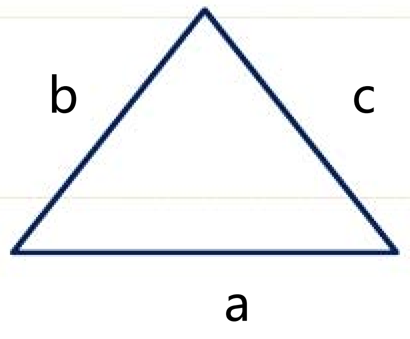
- What is the number of test cases for
  - Weak Normal Equivalence Class Testing
  - Strong Normal Equivalence Class Testing
  - Weak Robust Equivalence Class Testing
  - Strong Robust Equivalence Class Testing
  - Hybrid approach
- Given $x_1, x_2, x_3, \ldots, x_n$; valid ECs $E(x_1), E(x_2), \ldots, E(x_n)$; invalid ECs $I(x_1), I(x_2), \ldots, I(x_n)$:

- Given $x_1, x_2, x_3, \ldots, x_n$; valid ECs $E(x_1), E(x_2), \ldots, E(x_n)$; invalid ECs $I(x_1), I(x_2), \ldots, I(x_n)$:
  - Weak Normal Equivalence Class Testing
    - $\mathrm{Max}\{E(x_1), E(x_2), \ldots, E(x_n)\}$
  - Strong Normal Equivalence Class Testing
    - $E(x_1) * E(x_2) * \ldots * E(x_n)$
  - Weak Robust Equivalence Class Testing
    - $\mathrm{Max}\{E(x_1), E(x_2), \ldots, E(x_n)\} + \sum_{i=1}^{n} I(x_i)$
  - Strong Robust Equivalence Class Testing
    - $\prod_{i=1}^{n} (E(x_i) + I(x_i))$
  - Hybrid approach
    - $E(x_1) * E(x_2) * \ldots * E(x_n) + \sum_{i=1}^{n} I(x_i)$

Triangle Problem

Specification:

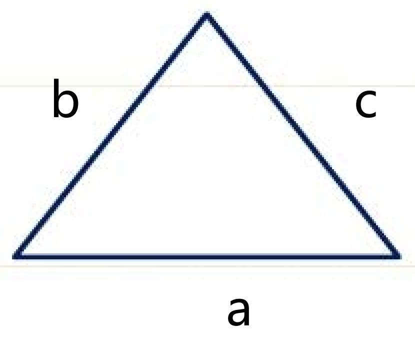
The triangle program accepts three integers: a, b, and c, as input. These are taken to be sides of a triangle. The output of the program is the type of triangle determined by the three sides: Equilateral, Isosceles, Scalene, or NotATriangle.



Triangle Problem

The triangle program accepts three integers, a, b, and c as the three sides of a triangle.The integers a, b, and c must satisfy the following conditions:

c1. $1 \le a \le 200$
c2. $1 \le b \le 200$
c3. $1 \le c \le 200$
c4. $a < b + c$
c5. $b < a + c$
c6. $c < a + b$



Approaches

1. Generate one ECs for each possible output (Equilateral,Isosceles,Scalene,or NotATriangle)

2. Generate ECs based on the relationship between the length of the sides.

Approach 1: EC based on output

Valid ECs:
V1 = {<a,b,c> : the triangle with sides a,b,and c is equilateral}
V2 = {<a,b,c> : the triangle with sides a,b,and c is isosceles}
V3 = {<a,b,c> : the triangle with sides a,b,and c is scalene}

Invalid ECs:
IV1 = {<a,b,c> : sides a,b,and c do not form a triangle}
IV2 = {<a,b,c>: a is not in the range of permitted values}
IV3 = {<a,b,c>: b is not in the range of permitted values}
IV4 = {<a,b,c>: c is not in the range of permitted values}

Robust EC test cases

Test Case	a	b	c	ExpectedOutput	EC
1	5	5	5	Equilateral	V1
2	2	2	3	Isosceles	V2
3	3	4	5	Scalene	V3
4	4	1	2	NotaTriangle	IV1
5	-1	5	5	Valueofaisnotintherangeofpermittedvalues	IV2
6	5	-5	6	Valueofbisnotintherangeofpermittedvalues	IV3
7	2	4	-3	Valueofcisnotintherangeofpermittedvalues	IV4

We have one test case for each type of output. Since we have only one output variable,strong/weak EC also generate the same set of test cases.

Are these test cases sufficient to reveal all possible faults?

A possible fault in implementation

Correct implementation:

```
if(a==b && b==c)
    output( "Equilateral" )
else if (a==b || b==c || a==c)
    output( "Isosceles" );
else
    output( "Scalene" );
```

Faulty implementation:

```
if(a==b && b==c)
    output( "Equilateral" )
else if (a==b)
    output( "Isosceles" );
else
    output( "Scalene" );
```

In order to reveal the fault,we should use 3 equivalence classes for the 3 pairs of equal sides in isosceles triangle!

How about IV1~IV4?

Approach 2: EC based on input (relationship between the sides)

We may also generate equivalence classes based on input,e.g.

$V1 = \{<a,b,c> : a = b = c\}$

$V2\_1 = \{<a,b,c> : a = b, a \neq c\}$

$V2\_2 = \{<a,b,c> : a = c, a \neq b\}$

$V2\_3 = \{<a,b,c> : b = c, a \neq b\}$

$V3 = \{<a,b,c> : a \neq b, a \neq c, b \neq c\}$

Additional ECs

We can identify additional equivalence class to test the triangle property

$$\left.\begin{array}{l} IV1\_1 = \{<a,b,c> : a \ge b + c\} \\ IV1\_2 = \{<a,b,c> : b \ge a + c\} \\ IV1\_3 = \{<a,b,c> : c \ge a + b\} \end{array}\right\} \text{Cannot form a valid triangle!}$$

We may further break IV1 into the following equivalence classes :

$IV1\_1' = \{<a, b, c>: a = b + c\}$

$IV1\_1'' = \{<a, b, c>: a > b + c\}$

Equivalence classes can be defined using the following guidelines:

- If an input condition specifies a range, one valid and two invalid equivalence classes are defined.
- If an input condition requires a specific value, one valid and one invalid equivalence classes are defined.
- If an input condition specifies a member of a set, one valid and one invalid equivalence classes are defined.
- If an input condition is Boolean, one valid and one invalid classes are defined.
- Recursively……

Exercises

- A program tries to calculate salaries for Professor, Associate professor, Lecturer, and assistant teacher respectively.
  - Valid ECs
  - Invalid ECs
- In Pascal: a sentence should be ended by ‘;’ .
  - Valid ECs
  - Invalid ECs

次日函数的有效值区间和无效等价类

- 有效值区间
  - M1＝{月份：1≤月份≤12}
  - D1＝{日期：1≤日期≤31}
  - Y1＝{年：1812≤年≤2012}
- 无效等价类
  - M2＝{月份：月份 < 1}
  - M3＝{月份：月份 > 12}
  - D2＝{日期：日期 < 1}
  - D3＝{日期：日期 > 31}
  - Y2＝{年：年 < 1812}
  - Y3＝{年：年 > 2012}

Exercise

- Design EC test cases for the NextDate Problem based on INPUT
  - Weak Normal Equivalence Class Testing
  - Strong Normal Equivalence Class Testing
  - Weak Robust Equivalence Class Testing
  - Strong Robust Equivalence Class Testing

次日函数的弱一般等价类测试用例

测试用例	月份	日期	年	预期输出
WN1，SN1	6	15	1912	1912年6月16日

- 强一般等价类测试用例与弱一般等价类测试用例相同

弱健壮测试用例的完整集合

用例 ID	月份	日期	年	预期输出
WR1	6	15	1912	1912年6月16日
WR2	-1	15	1912	月不在有效值域1..12中
WR3	13	15	1912	月不在有效值域1..12中
WR4	6	-1	1912	日不在有效值域1..31中
WR5	6	32	1912	日不在有效值域1..31中
WR6	6	15	1811	年不在有效值域1812..2012中
WR7	6	15	2013	年不在有效值域1812..2012中

额外强健壮等价类测试用例

三维立方体的一个“角”

共计多少条测试用例？补充完整

用例 ID	月份	日期	年	预期输出
SR1	-1	15	1912	月不在有效值域 1..12 中
SR2	6	-1	1912	日不在有效值域 1..31 中
SR3	6	15	1811	年不在有效值域 1812..2012 中
SR4	-1	-1	1912	月不在有效值域 1..12 中 日不在有效值域 1..31 中
SR5	6	-1	1811	日不在有效值域 1..31 中 年不在有效值域 1812..2012 中
SR6	-1	15	1811	月不在有效值域 1..12 中 年不在有效值域 1812..2012 中
SR7	-1	-1	1811	月不在有效值域 1..12 中 日不在有效值域 1..31 中 年不在有效值域 1812..2012 中

对日期进行处理的方法

- 如果它不是某个月的最后一天，则次日函数会直接对日期加1
- 到了月末，下一个日期是1，月份加1
- 到了年末，日期和月份都会复位到1，年加1
- 闰年问题要确定有关的月的最后一天

按照对日期进行处理的方法划分的等价类

- M1＝{月份：每月有30天}
- M2＝{月份：每月有31天}
- M3＝{月份：此月是2月}
- D1＝{日期：1≤日期≤28}
- D2＝{日期：日期＝29}
- D3＝{日期：日期＝30}
- D4＝{日期：日期＝31}
- Y1＝{年：年＝2000}
- Y2＝{年：年是闰年} 且年≠2000
- Y3＝{年：年是平年}

Recursively valid EC partitioning

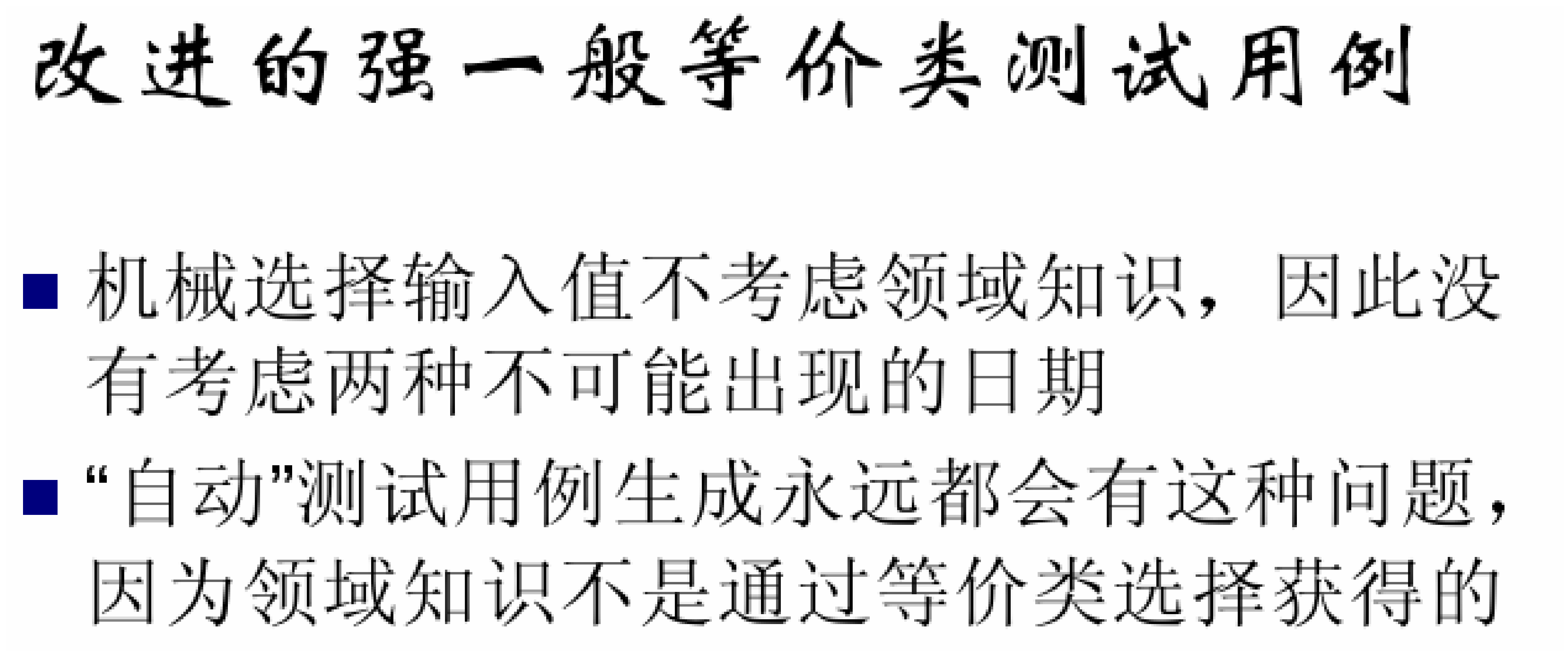
- How many test cases based on the new ECs？

Weak normal	4
Strong normal	3*4*3
Weak robust	4+2+2+2
Strong robust	5*6*5
Hybrid	3*4*3+2+2+2

按新的等价类划分方法产生的弱等价类测试用例

- 机械地从对应类的近似中间选择输入

用例 ID	月份	日期	年	预期输出
WN1	6	14	2000	2000年6月15日
WN2	7	29	1996	1996年7月29日
WN3	2	30	2002	2002年2月31日（不可能的日期）
WN4	6	31	2000	2000年7月1日（不可能的输入日期）



改进的强一般等价类测试用例

- 机械选择输入值不考虑领域知识，因此没有考虑两种不可能出现的日期
- “自动”测试用例生成永远都会有这种问题，因为领域知识不是通过等价类选择获得的

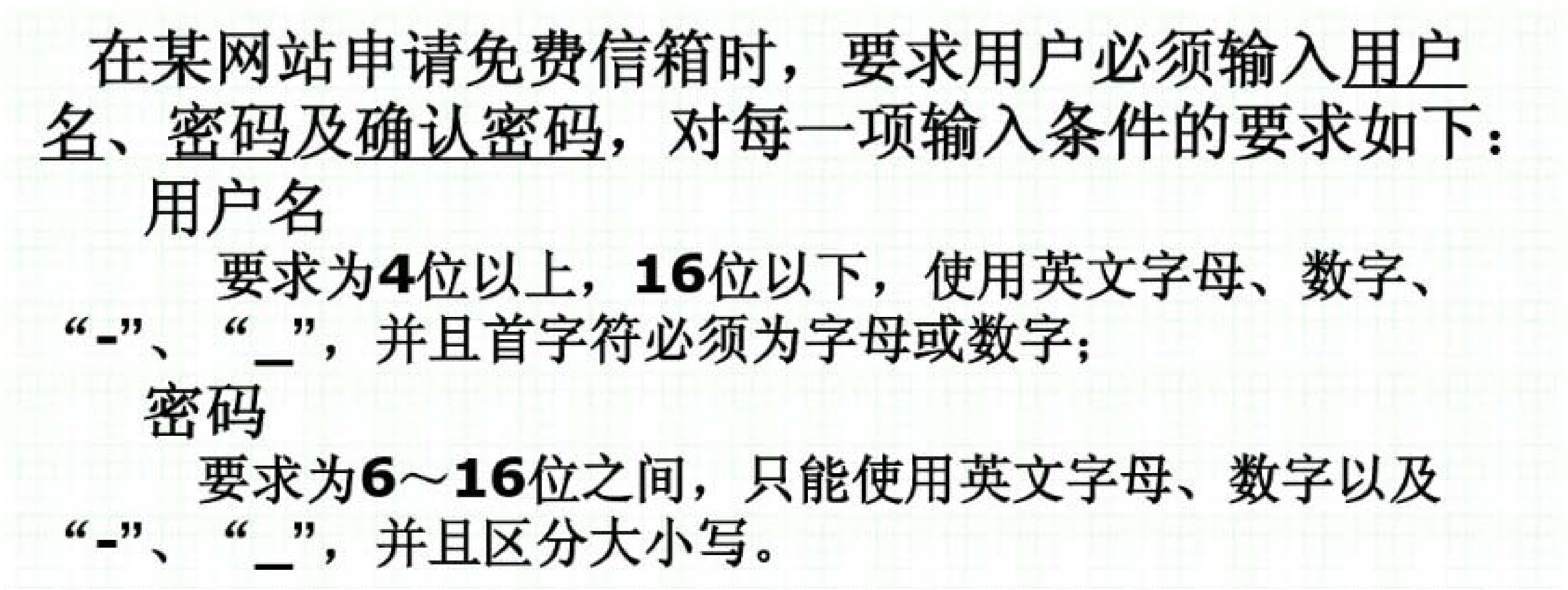
Infeasible combinations

Miss some important test cases

Exercise-1

- Try to use EP to design test cases for Commission Problem.
  - Input? Output?
  - Use BVA and EP together

Exercise-2



在某网站申请免费信箱时，要求用户必须输入用户名、密码及确认密码，对每一项输入条件的要求如下：

用户名

要求为4位以上，16位以下，使用英文字母、数字、“-”、“_”，并且首字符必须为字母或数字；

密码

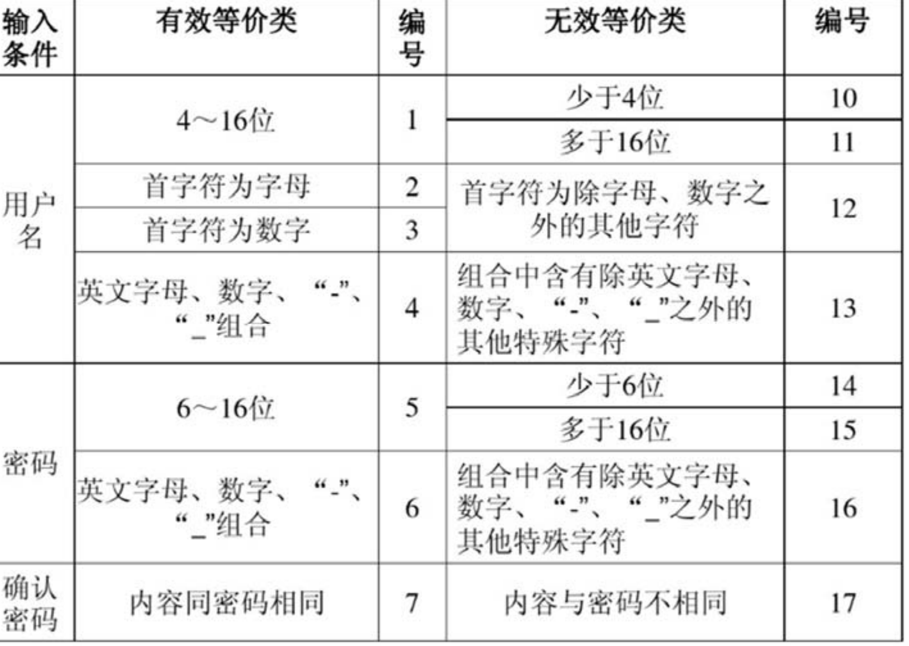
要求为6～16位之间，只能使用英文字母、数字以及“-”、“_”，并且区分大小写。

给出有效等价类和无效等价类，并设计测试用例

Exercise-2

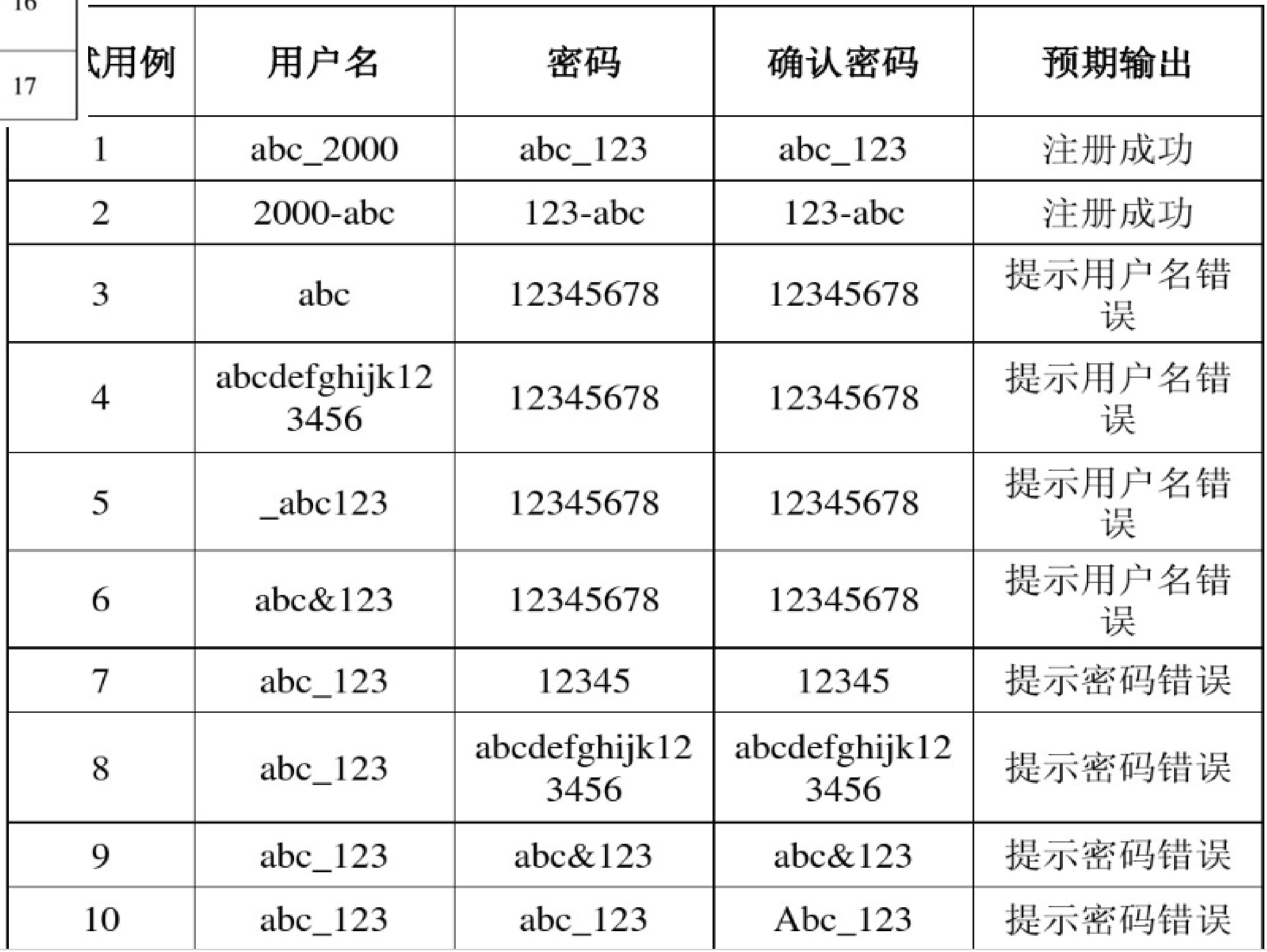
输入条件	有效等价类	编号	无效等价类	编号
用户名	4～16位	1	少于4位	10
			多于16位	11
	首字符为字母	2	首字符为除字母、数字之外的其他字符	12
	首字符为数字	3		
	英文字母、数字、“-”、“_”组合	4	组合中含有除英文字母、数字、“-”、“_”之外的其他特殊字符	13
密码	6～16位	5	少于6位	14
			多于16位	15
	英文字母、数字、“-”、“_”组合	6	组合中含有除英文字母、数字、“-”、“_”之外的其他特殊字符	16
确认密码	内容同密码相同	7	内容与密码不相同	17

cise-2



输入条件	有效等价类	编号	无效等价类	编号
用户名	4～16位	1	少于4位	10
			多于16位	11
	首字符为字母	2	首字符为除字母、数字之外的其他字符	12
	首字符为数字	3		
	英文字母、数字、“-”、“_”组合	4	组合中含有除英文字母、数字、“-”、“_”之外的其他特殊字符	13
密码	6～16位	5	少于6位	14
			多于16位	15
	英文字母、数字、“-”、“_”组合	6	组合中含有除英文字母、数字、“-”、“_”之外的其他特殊字符	16
确认密码	内容同密码相同	7	内容与密码不相同	17

- 快速覆盖有效等价类
- 针对每个无效等价类分别设计测试用例



式用例	用户名	密码	确认密码	预期输出
1	abc_2000	abc_123	abc_123	注册成功
2	2000-abc	123-abc	123-abc	注册成功
3	abc	12345678	12345678	提示用户名错误
4	abcdefghijk123456	12345678	12345678	提示用户名错误
5	_abc123	12345678	12345678	提示用户名错误
6	abc&123	12345678	12345678	提示用户名错误
7	abc_123	12345	12345	提示密码错误
8	abc_123	abcdefghijk123456	abcdefghijk123456	提示密码错误
9	abc_123	abc&123	abc&123	提示密码错误
10	abc_123	abc_123	Abc_123	提示密码错误

Exercise-3

- 某城市的电话号码由三部分组成。这三部分的名称和内容分别是
  - 地区码：空白或三位数字；
  - 前　缀：非**’0’**或**’1’**开头的三位数；
  - 后　缀：四位数字。
- 假定被调试的程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的号码，就可用等价分类法来设计它的调试用例。

输入条件	有效等价类	无效等价类
地区码	空白(1), 3位数字(2)	有非数字字符(5), 少于3位数字(6), 多于三位数字(7)
前缀	从200到999之间的3位数字(3)	有非数字字符(8), 起始位为"0"(9), 起始位为"1"(10), 少于3位数字(11), 多于3位数字(12)
后缀	4位数字(4)	有非数字字符(13), 少于4位数字(14), 多于4位数字(15)

Exercise-3

- 4个有效等价类，可以公用以下两个次数用例：（1）、（2）中各取一个对应合法的（3）、（4）即可。
- 对11个无效等价类，要选择11个调试用例

测试数据	范围	期望结果
(20A) 123－4567	无效等价类(5)	无效
(33) 234－5678	无效等价类(6)	无效
(7777) 345－6789	无效等价类(7)	无效
(777) 34A－6789	无效等价类(8)	无效
(234) 045－6789	无效等价类(9)	无效
(777) 145－6789	无效等价类(10)	无效
(777) 34－6789	无效等价类(11)	无效
(777) 2345－6789	无效等价类(12)	无效
(777) 345－678A	无效等价类(13)	无效
(777) 345－678	无效等价类(14)	无效
(777) 345－56789	无效等价类(15)	无效

等价类划分和边界值的区别与联系

- 示例：整型参数输入“1~99”为合法
  - 使用边界值设计测试用例
  - 使用等价类划分设计测试用例

- 边界值
- 0，1，2，50，98，99，100

等价类划分法

- 1. 按输入类型划分：数字、字母、符号等等；区分有效等价类/无效等价类
- 2. 在有效等价类中，按照位数不同划分：输入“空、个位数、十位数、百位数”
  - 测试了“空、1、11、101”，就不用在测试“2、22、102”了，因为“个十百”位数都是等价的，属于同一类型；

异同

- 等价类划分法：将测试过程中的输入、输出、操作等相似内容分组，从每组中挑选具有代表性的内容作为测试用例，划分有效等价类和无效等价类
- 边界值分析法：确认输入、输出的边界，然后取刚好等于、大于、小于边界的参数作为测试用例测试；
- 联系：等价类划分和边界值要一起考虑，边界值分析法属于等价类划分法的补充，任何等价区间都有边界，有边界就有等价区间。

Exercise-4

公园门票规定：

- 身高1.2m以下的儿童免票；
- 身高1.2～1.4m的儿童半票（含1.2m）；
- 年龄在60～69岁之间的老人半票（含60岁）；
- 年龄在70岁以上的老人免票（含70岁）；
- 在校学生半票（不含在职学生、电大学生）；
- 革命烈士家属、现役军人免票。

结合等价类划分和边界值分析法，设计测试用例。

Black Box Testing Techniques

- Boundary Value Analysis
- Equivalence Partitioning
- Decision Table
- Cause-effect Graph
- Combinatorial Test